EPARTMENT OF APPLIED ECONOMICS

RESEARCH REPORT

EXTENDED ENTERPRISE ARCHITECTURE WITH THE FADEE

FRANK GOETHALS • JACQUES VANDENBULCKE •
WILFRIED LEMAHIEU • MONIQUE SNOECK • BJORN CUMPS

OR 0424

EXTENDED ENTERPRISE ARCHITECTURE WITH THE FADEE[i]

Frank Goethals[1], Jacques Vandenbulcke[1], Wilfried Lemahieu, Monique Snoeck, Bjorn Cumps
F.E.T.E.W. – K.U.Leuven – Naamsestraat 69, B-3000 Leuven, Belgium
{Frank.Goethals, Jacques.Vandenbulcke, Wilfried.Lemahieu, Monique.Snoeck}@econ.kuleuven.ac.be
[1]SAP-leerstoel Extended Enterprise Infrastructures

Keywords: Coordination problems, Enterprise Architecture, Business-to-Business integration, Extended Enterprise, Extended Enterprise Architecture Framework, FADEE

Abstract: Business-to-Business integration (B2Bi) is considered to be not merely an IT-issue, but also a business problem. This paper draws attention to the challenges companies within an Extended Enterprise are confronted with when integrating their systems. We primarily pay attention to coordination problems that may arise. To overcome these problems we propose the use of Enterprise Architecture descriptions. We discuss the powers of using Enterprise Architecture descriptions in integration exercises. It will become clear that doing Enterprise Architecture is no longer an option; it is mandatory. Furthermore, we present the FADEE, the Framework for the Architectural Description of the Extended Enterprise. This framework gives an overview of how companies can apply the Zachman framework to do Enterprise Architecture in the realm of the Extended Enterprise.

1. INTRODUCTION

Nowadays companies are offering Web services (i.e. information system services) to other companies, and are using Web services offered by other companies. The business and IT landscapes have turned more complex than ever, and the creation and automation of processes that involve services of different companies is an evolutionary challenge. In the past, many IT projects have failed, and many will fail in the future if no better way is found to handle these projects. Unfortunately, partnerships can be harmed if the envisioned IT integration projects between partners fail. In this article, we propose the use of architecture descriptions as a means to support the integration of systems at a Business-to-Business (B2B) level, and more specifically at the level of the Extended Enterprise (see below). In what follows we first structure the B2B domain, and set forth basic observations concerning B2B integration (B2Bi) practices we should keep in mind when searching for a good solution to the integration problem. Next, in Section 3, we define the communication problems that arise when developing Web services for the Extended Enterprise; and in Section 4 we discuss the idea of Extended Enterprise Architecture Descriptions as a means to solve the communication problems. Finally, in Section 5, we introduce the Framework for the Architectural Description of the Extended Enterprise.

2. STRUCTURING THE B2B DOMAIN

From the theory on the network form of organizations (see e.g. (Podolny and Page, 1998)), it is clear that companies may be involved in an organizational integration at three levels, namely

- at the level of the *individual enterprise* the different departments have to be integrated,

- at the level of the *Extended Enterprise* the companies that make up the Extended Enterprise have to be integrated. By the term Extended Enterprise (EE), we mean *a collection of legal entities (N ≥ 2) that pursue repeated, enduring exchange relations with one another.*
- at the level of the *market* a very loose coupling is present with companies in the environment (other than those within the Extended Enterprise). With these companies no long term relationship is envisioned.

It is remarkable that an Extended Enterprise truly forms a new enterprise that has a starting point and an endpoint (in time). Consequently, this new (extended) enterprise can (and should) be architected by a group of people (including CEO and CIO) of the partnering companies! This is in contrast to doing business in the marketplace, where transactions happen at isolated moments in time and no new enterprise is formed.

Contingency theory reveals that there should be a fit between an organization's structure, its technology, and the requirements of its environment (see e.g. (Borgatti, 2001)). As companies within an EE face two types of environments (organizations within the EE vs. organizations outside the EE), they need appropriate IT approaches to deal with each type of environment. Consequently, we may say that companies are confronted with three types of information systems integration. Firstly, companies have to deal with the integration of their internal systems (Enterprise Application Integration, EAI). Stovepiped systems – often made to fit the requirements of one department – need to be integrated. Secondly, there is an integration with systems of other companies within the EE. We refer to this as EEi (EE integration). Thirdly, companies may want to integrate their systems with those belonging to other companies than close partners. We call this Market B2Bi. The three types are represented in Figure 1. In this figure, it is assumed that company A is forming an Extended Enterprise with companies B, C, D, E, and F; and is doing market transactions with companies G, H, I, J, K, and L.

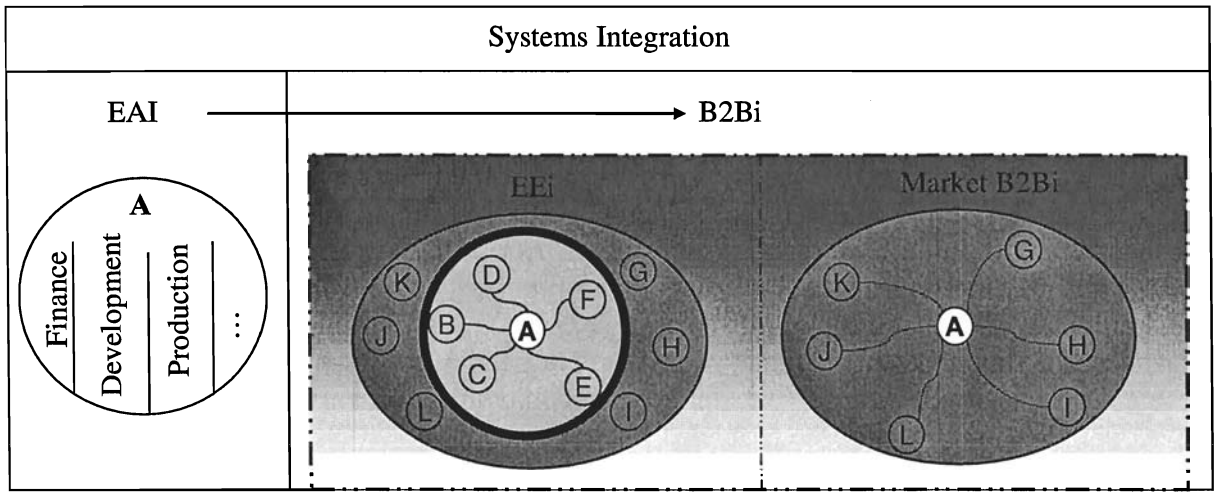

Figure 1: Three types of systems integration

The three types of integration each have their own specific issues. An important difference between EEi and Market B2Bi for example is that the human link between the companies is much less substantial for the latter. Also, Market B2Bi may include using Web services offered by parties that were unknown upfront, implying there should be a way to find the parties and the desired Web services. Unfortunately, the difference between these two forms of B2Bi is usually neglected in literature on IT.

For each of the three types of integration, we witness/foresee an evolution from static integration to more flexible, dynamic forms of integration. At the level of collaborating companies, the (relatively new) Web services paradigm is more flexible than (the older) EDI

(Electronic Data Interchange) technology. Also, in the future B2Bi could be enhanced by software agents that are capable of searching and binding Web services autonomously. Please note that, as is indicated by the arrow in Figure 1, EAI should precede B2Bi (see e.g. Linthicum (2000)).
One point that should be kept in mind when integrating businesses is that doing business is still about people's requirements, not just about IT. Note that in the commodity goods market, companies are not just offering goods without investigating which goods the consumers exactly want. Companies should become consumer-oriented in the Web services domain too, i.e., companies should research which Web services are interesting for business people from other companies rather than simply offering the services their own IT department deems useful.
In the remainder of this paper, we concentrate on one form of integration, namely Extended Enterprise integration (EEi).

3. REVEALING THE COMMUNICATION PROBLEMS

Many Web service integration challenges stem from communication problems (Goethals et al., 2003). In what follows we focus on two issues. First we propose challenges related to the concept of *consumer-oriented* Web services. Next, we discuss the idea of Web services *choreographies* to show how important communication can be.

3.1 The Quest for Consumer-Oriented Web Services Reveals Two Communication Gaps

Nowadays, companies are offering Web services to partners and other parties. When developing Web services, it is important to know the functional and non-functional requirements of the future service consumer. However, in actual practice the attention seems to be going much more to *playing* with Web services technology than to using the new technology in *a way interesting* to businesses (Frankel and Parodi, 2002).
In realizing consumer-oriented Web services many problems may arise. For many years, the problem of business-ICT alignment has challenged companies. Nowadays, an extra gap arises besides the one between business and ICT; namely the one between the different companies in an EE[ii]. Pollock (2002) states that *most problems contributing to the high failure rates of integration projects are not technical in nature.* Pollock points out the importance of semantics in B2Bi. While misunderstandings (and semantic obscurities) within a company may be large, the problems only increase when looking at relationships between different companies. Please note that the gap is not only present at business level, but also at IT-level. A database (DB) in one company may for example use the term 'customerno' to denote the same concept as 'customernumber' in the DB of another enterprise.
We conclude that there are two communication gaps. The problem is illustrated in Figure 2, the dotted lines show the communication gaps.

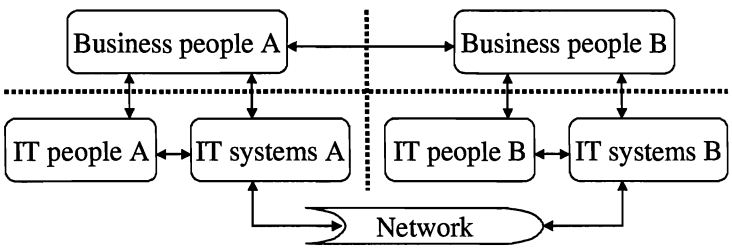

Figure 2: Two gaps in realizing B2Bi

Collaboration implies communication. Much communication can be automated (e.g. sending purchase orders), but communication at a meta-level, i.e., communication about the communication, is hard –if not impossible– to automate. As we will see, this level of (human) communication can be *supported* by architectural descriptions.

3.2 Creativity Requires Communication among Partners

One of the most-promising challenges in the B2B domain is the offering of Web services with a coarse-grained functionality, i.e., services that are composed of several other services. These smaller services are then called in parallel or in sequence and the call may be contingent on some conditions. Note that the *big* service may use *small* services of different companies. It is interesting to note that due to the ubiquity of the Internet and the SOAP standard companies with an EDI network have lost the competitive advantage of having automated communication, as Web services form a (cheaper) alternative that is available to everyone (i.e., the automation of standard processes becomes a commodity). Competition has shifted to a higher level: use the standards (such as TCP/IP, SOAP, and WSDL) *creatively* to realize new business practices so as to create a competitive advantage for the company!
Currently, Web services technology is mostly used for information exchange. However, if the Web services paradigm is to be *the* paradigm for B2Bi, it should also allow for the realisation of business transactions (all-or-nothing scenarios). Realising transactions in a B2B context can get very complicated. For one thing, the use of classic locking-protocols is not always realistic, as companies do not like other companies to have a lock on their data and as the completion of transactions might take quite some time (resulting in so-called 'long running' or 'long-lived' transactions).
Currently, much research is being done towards the realisation of transactions in a B2B context. Many kinds of structures are possible for realising transactions, depending on different degrees of trust, human relations, etcetera. We can illustrate this with a simple example (see Figure 3).

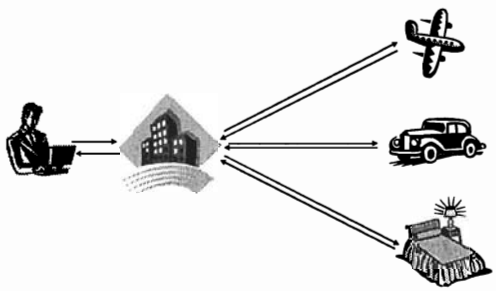

Figure 3: Realizing a transaction in the EE

First, imagine a travel agency offering tourists a *BookPlaneCarAndHotelWebservice*, which books an airplane seat, a car *and* a hotel room, or none of them. Upon a call of a traveller, the travel agency would contact the three relevant partners: an airplane booking company, a car rental company and a hotel booking company. Availability of airplane seats and cars may be confirmed immediately while the confirmation of the hotel booking company may keep the travel agency waiting for 24 hours. The consequence of this is that the travel agency needs the possibility to make reservations in the systems of the airplane booking company and of the car rental company! These reservations would be confirmed or cancelled when the reply of the hotel booking company arrives. This scenario clearly requires an outstanding relationship between the companies.
There is, however, a more realistic though less intuitive solution to the problem which requires less trust and could be the basis for more dynamic B2Bi. This scenario is shown in

Figure 4. The travel agency could ask the airplane booking company to reserve an airplane seat *and* to search for a car and a hotel room if an airplane seat was available. In this scenario, the airplane booking company can make the seat reservation herself (so the travel agency does not need to make reservations in the airplane booking company's systems!) and sends a request to the car rental company to book a car and to search for a hotel room. If the car rental company has a car available, she reserves this car herself and contacts the hotel booking company. The latter sends a confirmation or a denial to the car rental company, which confirms or cancels her own reservation and informs the airplane booking company of the result of the process. The latter then takes appropriate actions and informs the travel agency of the result[iii]. This whole process boils down to serializing the transaction process from Figure 3. This way, companies only make reservations in their own systems, and wait for the reply from the company downstream to decide whether the reservation should be confirmed or not.

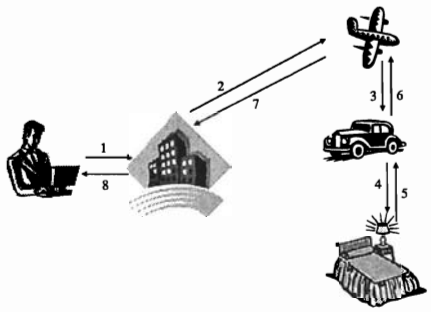

Figure 4: Realizing transactions in another way

It is clear that the combination of both presented structures offers possibilities for building bigger, more value-adding services. While standardization is very important and interesting at technical level (e.g. exchanging SOAP documents), creativity remains important when looking from a business perspective. Extended Enterprises should not straightforwardly automate existing public processes, but have to rethink them first! Companies can only get a competitive advantage by doing things other companies are not doing. Creativity combined with communication (among the right persons, such as CIOs) is indispensable to detect ways to apply ICT in a company to get advantages over competitors.

We conclude that companies (within an Extended Enterprise) want to offer useful services to each other through their IT-systems, but that people find themselves confronted with communication difficulties. Communication about the services that should be provided, and about the way they should be provided is very important, as new business practices and problems may only be revealed by discussing the issue. In our vision, the solution to the communication problem lies in offering every person the information he/she needs for doing his/her part of the B2Bi job, and mapping this information for different persons. Above this, the information should be made persistent and accessible. All this is exactly what we intend to do with architectural descriptions.

4. RESOLVING THE COMMUNICATION PROBLEMS WITH ARCHITECTURE DESCRIPTIONS

In what follows, we first discuss the idea of architecture descriptions (ADs) of software-intensive systems. Subsequently, we investigate how architecture descriptions could be of help in a B2B integration exercise (i.e., what is their power in that situation). In the next

section (Section 5), we will introduce a framework for Enterprise Architecture we have developed specifically for Extended Enterprise integration practices.

4.1 Introduction to Enterprise Architecture Descriptions

As stated, the Extended Enterprise *is* an enterprise and can as such be architected. The Generalised Enterprise Reference Architecture and Methodology (GERAM; IFIP-IFAC Task Force, 1999) presents a generic view of the lifecycle phases enterprises go through. The Zachman framework presented in this section can be mapped to the GERAM, as the Zachman framework can be seen as a kind of categorization of the models that can be built of an enterprise during the different life cycle phases of an enterprise. A mapping between the Zachman framework and GERAM is presented in (Bernus et al., 2003).
Zachman (1987), who is considered to be a pioneer in the realm of information systems architectures and Enterprise Architecture, discusses information system design by analogy to the work steps and the representations of the classical (building) architect and producers of complex engineering products. The Zachman framework relies on the fact that the description of something depends on the perspective from which you look at it, and on the question that was in mind when making the description. As such, the Zachman framework (as depicted in Figure 5) presents two dimensions along which architecture descriptions could be categorized. The first dimension (the succession of the rows in the figure) concerns the different perspectives of the different participants in the systems development process (the owner's view, the designer's view, the builder's view, etc.). By walking through the rows, one translates the business system into the IT system. Therefore, the enterprise as seen by the business people should fit (be aligned with) the ICT systems and vice versa. As such, one integrated, aligned enterprise is created. The second dimension (the sequence of the columns in the figure) deals with the six primitive English questions *what*, *how*, *where*, *who*, *when* and *why*. These are six aspects that should be considered when designing an enterprise (and her ICT systems). It is clear that there is not just *one* possible architecture description, but a *set* of architecture descriptions (ADs) that are additive and complementary. Changes in the architecture are likely to affect multiple architecture descriptions.

	Data (What)	**Function (How)**	**Network (Where)**	**People (Who)**	**Time (When)**	**Motivation (Why)**
Scope (Ballpark View)	List of things important to the business	List of processes the business performs	List of locations in which the business operates	List of organizations important to the business	List of events / cycles significant to the business	List of business goals / strategies
Business model (Owner's view)	e.g. semantic model	e.g. business process model	e.g. business logistics system	e.g. work flow model	e.g. master schedule	e.g. business plan
System Model (Designer's view)	e.g. logical data model	e.g. application architecture	e.g. distributed system architecture	e.g. human interface architecture	e.g. processing structure	e.g. business rule model
Technology Model (Builder's view)	e.g. physical data model	e.g. system design	e.g. technology architecture	e.g. presentation architecture	e.g. control structure	e.g. rule design
Detailed Representations (Subcontractor)	e.g. data definition	e.g. program	e.g. network architecture	e.g. security architecture	e.g. timing definition	e.g. rule specification

Figure 5: The Zachman Framework (Zachman, 1987)

The Zachman framework can capture all decisions that have to be made during the systems development process. Communicating these decisions to the relevant persons is essential. Decisions form constraints that have to be respected. It is clear that if persons are not aware of the constraints (e.g. because decisions were not communicated to them or because decisions have been made too long ago), they are taking uninformed decisions. It does not make any sense to give people the freedom to neglect hard constraints (see e.g. Cook (1996)).

Since Zachman the idea behind ADs has evolved, producing the IEEE 1471-2000 standard on Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE 1471-2000 defines an 'architectural description' as *a collection of products to document an architecture*, whereas 'an architecture' is defined as *the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution* (Maier). Furthermore, a 'view' is defined as *a description of the entire system from the perspective of a set of related concerns.* As such, *a view is composed of one or more models* (Lassing et al., 2001). Other important Enterprise Architecture concepts have been defined in ISO-WD15704 (IFIP-IFAC Task Force, 1999).

Companies do not have to model all the cells in the Zachman framework. After all, an AD is not a goal *an sich*, but is a means to realise other goals. This idea is also reflected in IEEE-1471, and in ISO-WD15704. These state that the stakeholder concerns should be used to justify the views, i.e., they drive the viewpoint selection (Maier). Consequently, before arbitrarily drawing up an AD, one should know what the description will be used for (see Section 4.2).

4.2 The Power of Extended Enterprise Architecture Descriptions

Drawing up ADs is a big effort, requiring time, money and people. Consequently, investing in such practices should be justifiable, i.e., doing Enterprise Architecture should render substantial benefits. One interesting point to note is that ADs cannot only be useful for EEi, but also for EAI. Companies are focusing nowadays on EAI, and consequently drawing up ADs *now* could pay off two times: during the EAI effort now, and on the EEi exercise tomorrow. Of course, different types of integration may ask *partly* for different information (see Section 5).

By now it is clear that one complicating factor in EEi concerns the communication about functional and non-functional requirements, something that can hardly be automated (at this moment at least) with semantic markup and the like. The only way out is to give people an incentive to communicate and to *support* their communication, easing, improving, and speeding the negotiations between companies.

Architecture models can clearly offer support for semantics, by unambiguously defining all terms and their relationships at different levels of abstraction. Making a data thesaurus is in this vision not different from making any other architecture description of the system.

ADs are useful as a basis for discussion, which – in our opinion – yields advantages for diverse reasons:

- *Understanding the organization of the other party* is quite a difficult, though important task. By understanding other parties, new practices, procedures and opportunities can be revealed. This, however, requires someone who handles the complexity and oversees the total domain (at an appropriate level of abstraction). ADs are a good means to handle such complexity by making interesting abstractions. Above this, ADs can serve as the basis for a brainstorming-session.
- *Service Level Agreements (SLAs) could be negotiated* on the basis of the ADs. After all, formulating SLAs also requires a translation of business requirements into technical requirements and technical measures. Moreover, internal SLAs are often deployed in

order to manage the expectations of service users (see for example (Koch, 1998)). People all too often expect too much from IT, and this may also be the painful truth in an EE.

- An AD can be used to inform, guide and constrain decisions, especially those related to IT investments (CIO Council, 2001). ADs can be a facilitator for realizing B2Bi, as they ease the adaptation of the architecture. It is *easier to manage something you know well*! An AD contains much valuable *information for making decisions on investments and for system development*. Note that it is good practice to evaluate the proposed architecture before getting into development. Clements et al. (2002) state that, although architecture evaluation is almost never included as a standard part of any development process, evaluating the architecture upfront is an important and inexpensive task. By *making issues explicit* in an AD, *problems can be detected early on*. One should not be making implicit assumptions about functionality (especially not in the *global* economy, where customs may differ from partner to partner!). It is still very hard to test and validate choreographies of services. By discussing difficult issues upfront, many problems can be avoided. Moreover, the sooner problems are noticed in the software development process, the lower the costs of resolving them (Boehm, 1981).

Doing Enterprise Architecture can thus save money, especially in the long run. Clearly, building architecture descriptions slows down implementation projects. However, if the system has to be changed in the future (and there is no doubt that doing so will be necessary) the available architecture descriptions will be invaluable.

5. EXTENDED ENTERPRISE ARCHITECTURE WITH THE FADEE

In what follows, we discuss the Framework for the Architectural Description of the Extended Enterprise (FADEE). First, we present an example of an Extended Enterprise process. Next, the FADEE is presented; and in Section 5.3, the use of the FADEE is illustrated. We end the discussion on Extended Enterprise Architecture with a discussion on the future of Enterprise Architecture.

5.1 An Example of an Extended Enterprise Process

In order to illustrate the problem we are tackling, we present a very simple example of an Extended Enterprise (EE) process. The process is illustrated in Figure 6.

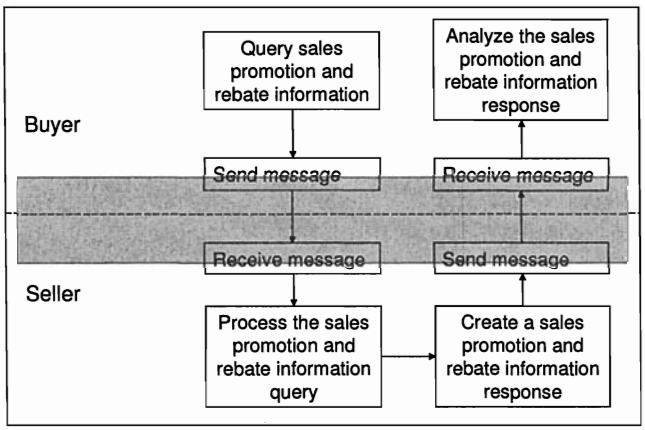

Figure 6: RosettaNet PIP2A4 Business Process Model

The example is based on PIP2A4 from RosettaNet: Query Sales Promotion & Rebate Information. This PIP is used by buyers to query the systems of suppliers to get information about sales promotions and rebates.
The question arises how we can document this process in ADs. First and for all we should decide whether a monolithic, centralized AD should be drawn which includes all of the descriptions of the systems in all of the enterprises making up the EE, or whether decentralized ADs should be drawn (i.e., ADs for each single enterprise), which are integratable and related to each other. The question is illustrated in Figure 7.

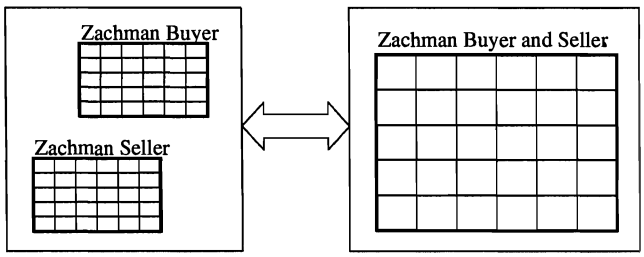

Figure 7: Decentralized vs. centralized ADs

5.2 The Framework for the Architectural Description of the Extended Enterprise

To answer this basic question we should think of several arguments. There are some reasons to believe *decentralized* ADs are desirable:

- Within the context of the EE, companies remain separate legal entities, keeping control over their own systems. EE partnerships are in contrast with mergers and acquisitions, which give rise to bigger legal entities. As companies enter and leave the EE, they are most likely not willing to put much effort in drawing up an EEAD (Extended Enterprise Architecture Description) that may become outdated fast. Note that companies find it hard to draw up ADs, even for their own isolated company. There is a chance that putting up a centralized EEAD becomes that big a job, that the flexibility to change the EE disappears.
- ADs can be used to better handle complexity by making abstractions. Why would one try to put all information in a centralized AD, if only part of this is relevant for each company? If there is no immediate contact between two companies in a business web, then they do not need to know each other's services.

Following these arguments, one could conclude that ADs of EEs should be decentralized descriptions. However, there is a reason to believe choosing for centralized ADs could be justified as well:

- Companies form an EE because the value of the whole is bigger than the sum of the values of the parts, at least at business level. This should be reflected in the ADs. We must face the fact that it would be very useful if the CIO and the top managers would know the whole picture, to overview the total process that evolves from the subparts. In our opinion, people like to have the feeling they are in control of something, and one only gets this feeling if he knows all information. Ambler (2002) states that *looking at the whole picture is the norm, not the exception* (but Ambler is not talking about *EE*ADs).

Consequently, a hybrid solution may be interesting. Such a hybrid solution would contain centralized ADs *and* decentralized ADs. Some other arguments supporting the hybrid alternative are the following:

- If decentralized ADs would be used, where would the connection between the private elements of every company be? The programs are sending messages, but whereto? This should be documented somewhere. This makes it tempting to include the service

descriptions of the services offered by partners into the ADs. However, it is illogical to include services from external parties into the description of your own systems. The Zachman framework – which was primarily designed for documenting one legal entity, one single enterprise – suggests modelling business processes performed by the enterprise, the breakdown of these processes, the IT infrastructure to realize these processes, etcetera. All of these are expected to be under the control of this individual organization. Sending messages between departments within one organization can be modelled in the business process model, but sending messages to the outside world causes part of the business process to be executed somewhere else. A company does not know much about the IT infrastructure behind the external services she is calling: she only needs to know *where* to send *which* SOAP (Simple Object Access Protocol) messages, and which SOAP messages could be expected in reply. Above this, the process that is executed by the other party may be very complicated (e.g., because interaction is needed with some other parties in turn), and therefore it is not necessary to know all the details of this business process either. Please note that a loose coupling between companies/services requires service updates to be transparent, i.e., the service requestor should not notice the way the service is being realized; and changes to the way the service is performed should be transparent.

- During the last few years, the concept of federated enterprise architecture descriptions received more and more attention from the American government. Two (for our purposes) interesting frameworks were developed: the Federal Enterprise Architecture Framework (FEAF, (CIO Council, 1999)), and the Treasury Enterprise Architecture Framework (TEAF, (Department of the Treasury, Chief Information Officer Council, 2000)). These frameworks were developed to come to integratable ADs. The FEAF and the TEAF are both very similar to the Zachman framework (which is still *the* reference when it comes to architecture). They suggest that every agency independently draws up a number of architecture descriptions, so as to describe at least the *what*, *how*, and *where* (and *who* in the case of TEAF) from different perspectives. The TEAF even proposes a number of specific models that should be drawn up to populate this 'small' Zachman framework. It is interesting to note that it was assumed that no monolithic, centralized architecture description would be needed to come to an integrated view on the architecture. In both cases, the 'segment architecture' approach was taken. *A segment is considered to be an enterprise within the total Federal Enterprise.* This approach would allow critical parts (architectural segments) of the overall federal enterprise to be developed individually within a structured Enterprise Architecture framework. During the last years, however, it was noticed that no cross-agency integration could be realized without some central documentation (and coordination). Therefore, a new framework, the FEA (Federal Enterprise Architecture) was developed, that functions as a cross-agency classification schema of business and IT practices. Please note that the FEA (which is still under development) cannot be regarded as a true Enterprise Architecture framework: the framework is not meant to hold *models*: it just categorizes initiatives (and as such it is very different from the FEAF and the TEAF). Please keep in mind that the US government is one legal entity, whereas companies within an EE remain separate legal entities.
- Using a Zachman framework at two levels (centralized and decentralized) implies that we actually have two definitions of what constitutes the 'enterprise'. The definition of the scope of the enterprise is very important as this is the level at which 'stovepipes' are being created. In the past, IT served different departments within an organization too much as isolated entities, creating stovepipes with a size smaller than that of the legal entity. The result was a disintegrated enterprise. Stovepipes can internally be tightly integrated, but should be loosely coupled to each other. Choosing the scope of the enterprise implies

choosing which level of integration can be realized. As the Extended Enterprise is no single legal entity, and companies really want to realize a loose coupling between their systems (via XML Web services for example) it is logical to see the legal entity as the scope of the enterprise. However, if the goal is to integrate systems of different companies, a view is needed from the enterprise where the enterprise is the collection of collaborating companies.

- Each enterprise (thus also the Extended Enterprise) has an *enterprise life cycle* (see the GERAM; IFIP-IFAC Task Force, 1999). As every enterprise needs to be engineered, it is logical to draw up ADs for all enterprises.
- The idea to draw up two levels of ADs dovetails with Drucker's management concept of federal decentralization. Cook (1996) asserts that a successful implementation of distributed computing is only possible if this concept of federal decentralization is respected. This means that both are needed, a strong center (the level of the EE), and strong parts (the level of the individual enterprises). This way, the flexibility of decentralized computing can be combined with the coordination advantages of centralization.

Consequently, it is clear that a hybrid solution is needed: the architectural descriptions of systems should be split into two parts, namely the documentation of the internal processes, data, systems, etcetera; (the 'individual enterprise AD') and the documentation of the practices that relate to the EE realization (the 'EEAD'). For both parts, a separate Zachman framework could be used to categorize the descriptive artifacts.

The EEAD thus contains all processes which include services (at least 1) that are executed by other parties. However, *how* these services (the subtasks of the total process) are being realized is not mentioned in the EEAD. The parts of the process that are being realized by the company itself are documented in the individual enterprise AD, the services that are executed by other parties are considered to be black boxes, and no further information concerning these services is noted down. The two types of ADs are brought together in what we call the FADEE: the Framework for the Architectural Description of the Extended Enterprise.

5.3 The FADEE Illustrated

Let us define an 'ad-hoc purchasing process', which includes RosettaNet PIP2A4. This process is depicted in Figure 8. The ad-hoc purchasing process is different from the classical purchasing process in that the company does not initiate an order because the stocks are low, but because of the fact that suppliers are offering sales promotions, and the stocks are low enough to allow the company to accept the attracting offer of the supplier. The ad-hoc purchasing process is nothing more than a sequence of some subtasks: first the suppliers are asked for their special offers (external service), next the PO (purchase order) quantities are determined (internally), and finally the purchase order is placed with the supplier (external service).

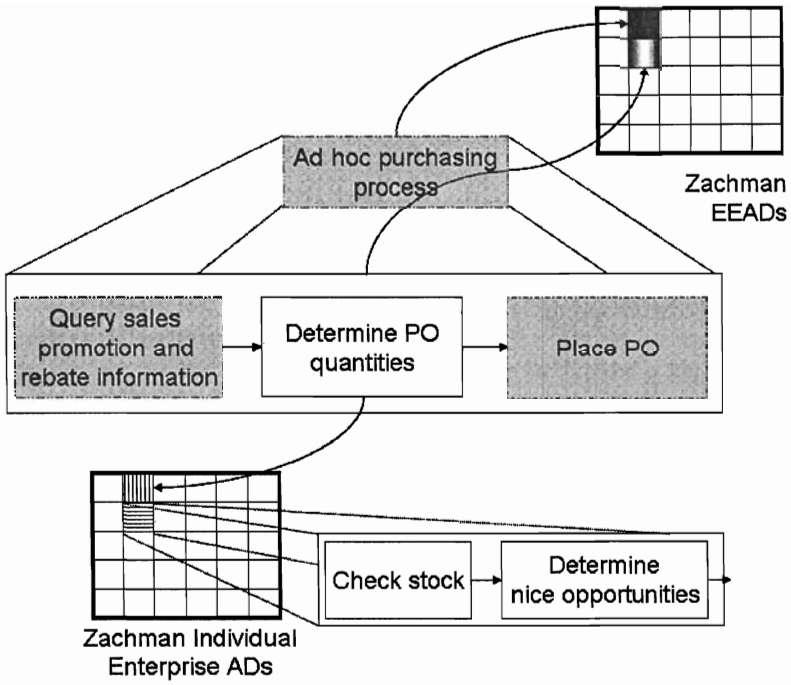

Figure 8: The FADEE illustrated

It is clear that the process includes services of external parties, so the process is first documented in the EEAD. Next, this process is subdivided into its subtasks. In the EEAD, all these subtasks are regarded as black boxes. The EEAD contains the messages that are to be sent between subtasks, and references to the locations of the services. One of the subtasks, the second one, is executed internally. This subtask, determining the PO quantities, is thus documented in the individual enterprise AD.

The example we just gave was a very simple one, in which the process that was documented in the EEAD was not a service that was to be made accessible over the Internet. However, we may also use the FADEE to document coarse-grained processes which are accessible over the Internet (as a Web service). Imagine three companies, an airplane booking company (A in Figure 9), a hotel booking company (B in Figure 9), and a car rental company (C in Figure 9). These companies may want to offer a BookHotelAirplaneAndCarService by bundling smaller services they offer separately. This coarse-grained service only exists at the level of the EE, it does not exist at the level of the separate legal entities. The choreography of the services (including the messages that would have to be sent between the different parties) could be described in the EEADs, while each company would document its own, smaller services in its individual ADs.

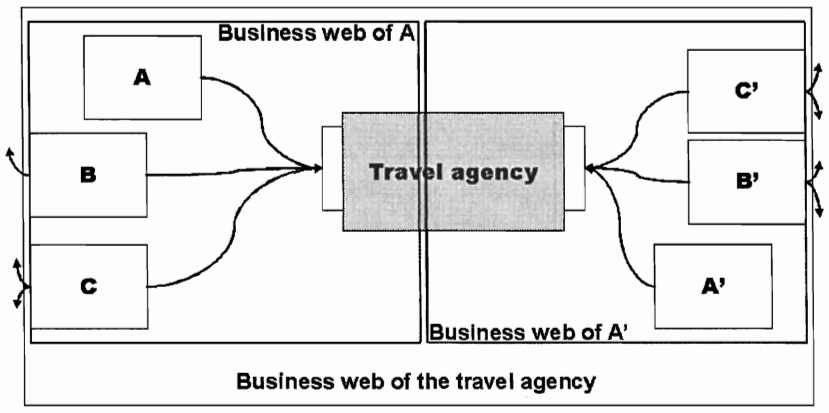

Figure 9: Example of a BookHotelAirplaneAndCarService

Now, imagine that these three companies have got an agreement with the travel agency that states that the A, B and C network is only allowed to offer this Web service to this specific travel agency. The travel agency, on the other hand, may have contacts with several networks that offer similar BookHotelAirplaneAndCarServices (for example a network formed by companies A', B' and C', as depicted in Figure 9). The EE from the point of view of the travel agency would be different from the EE from the point of view of company A (which in turn could be different from the EE from the point of view of company B, which may have partnerships with other companies).
Most of the information in the EEAD of the travel agency is irrelevant for company A; and above all, the travel agency most probably wants to keep most of this EEADs secret. Also, the three cooperating companies A, B, and C probably do not want the travel agency to know how the service is delivered, and the travel agency does not really need to know this. In short, some Extended Enterprises may have parts in common, but still they need to build an EEAD of their own.
The rationale we have just followed for processes could also be used for the other columns of the Zachman framework. In fact, if we apply this idea to the data-column, we notice that this idea is relatable to the concept of using ontologies and a data thesaurus. Companies could keep on using their own terminology internally, and map this terminology (made explicit in a data thesaurus) to a standard-terminology (the ontology) via the data-column of the EEAD. Of course, some data may be defined at the level of the EE that does not exist at the level of the separate enterprises (e.g., the number of trips that were sold consisting out of a hotel room, an airplane seat, and a rented car).
Companies will have to find out themselves which columns are the most important ones in their case. If we follow the FEAF and the TEAF, it is clear that the when and why columns (and the who column) are often considered to be of minor importance. In any case, the total FADEE offers all models of the enterprise that can be useful to decide on the future of the enterprise[iv].

5.4 The Future of Architecture Descriptions

In Section 4.2 we discussed some advantages of using ADs in an EEi setting. These advantages could be materialized today.
Now that we have presented how these ADs could look like, we should consider future applications of ADs. In what follows, we motivate why we believe that the concept of ADs could become even more powerful than what we have presented in this paper. This should be considered as a call to diverse organizations in the modeling community to integrate their efforts.
Architectures can be described in appropriate XML-variants (which are all grouped under the name 'Architecture Markup Languages', AMLs). The Architecture Description Markup Language (ADML) is an XML-based representation language that was specified to provide interoperability of architecture information, both between architecture tools and throughout the systems lifecycle (The Open Group, 2003). The ADML is used through tools, i.e., enterprise architects do not use the ADML directly. It is interesting to note that there is a dialog going on between the ADML community and the UML community, which could improve the interoperability between tools throughout the systems lifecycle[v]. Hilliard states that the ADML is insufficient to capture a multi-viewpoint architecture description (in the sense of IEEE 1471). Therefore, another architecture markup language, the Markup Language for Architectural Description (MLAD) is being developed, based on IEEE 1471.
There are many evolutions going on in the IT domain. All of these seem to ask for isolated solutions: there are specific languages for describing choreographies of Web services

(BPEL4WS, BPML, etcetera), specific markup languages (e.g., RDF and DAML) are being developed to shape the semantic web, other efforts are aimed at describing and finding Web services (e.g. WSDL, UDDI and DAML-S), etcetera. We should, however, make sure to get an integrated B2Bi solution. If people start using ADs detached from choreography description documents and semantic markup, the complexity of the total solution will increase. We should try to solve different problems at a time!
The architectural description of processes (in an AML) as suggested in this paper is closely related to Web services choreography languages as the BPEL4WS (Business Process Execution Language for Web Services) and the BPML (Business Process Modeling Language).
By describing the architecture in a semantic web language (such as RDF, DAML-S and the like), one can allow computers to search the capabilities of the described system by navigating through the ADs. Consequently architecture markup language efforts should not be seen isolated from classic semantic web efforts. EEADs may be seen as an extra, semantic layer on top of the existing system. This idea could give rise to a non-intrusive solution to the problem of semantics.
Also, if (mature) code generators one day find their way to the market, they can take the AML documents as an input. Frankel and Parodi (2002) state that the MDA (Model-Driven Architecture) *sets the stage for automatic generation of at least part of the XML and code, such as Java code, that implements the* [web] *services*. The MDA uses UML to specify services precisely and in a technology-independent manner. Of course, code generation demands very detailed descriptions, while ADs in general do not need to be that detailed (the measure of detail of the description should reflect the goal of the AD). Consequently, AMLs should allow the iterative elaboration of ADs, allowing the detailing of existing models when code generation becomes a real opportunity. Please note that Ambler (2002) states it is good practice to work iteratively and incrementally with ADs (even if code generation is not an issue). There has been written a lot about code generation, but not much has been realised yet. Nevertheless, it looks like the future will bring improved code generation tools. In our opinion, this is a nice prospect as creativity should show in the architecture of the IT-system, rather than in the code as such.

6. CONCLUSIONS

We have identified the communication problems that exist in the Web services world, and we have proposed a means to solve this problem. While ADs have been used in the past in the context of separate legal entities, they now also seem interesting in the case of B2Bi.
Providing Web services is not just an IT topic, but *also* a business matter. The design of Web services requires a lot of communication between persons with different backgrounds, capacity and vocabularies. To support this communication, architecture descriptions are very helpful. Above this, it is clear that documenting IT systems is a very important prerequisite to come to a manageable and maintainable IT infrastructure. Zachman stresses that achieving alignment, flexibility, integration, and reusability is only possible when the enterprise is being architected. In the future, code may be generated from the models that describe the system; and more dynamic forms of B2B integration could be based on architecture descriptions represented in an Architecture Markup Language. We conclude that architecture descriptions are invaluable, especially in the case of B2Bi. We hope this paper draws the attention to the value of Enterprise Architecture in the B2B domain. Companies should balance their short term goals (fast implementation) with a long term vision (flexibility, integration, ...).

Enterprise Architecture is *the* way to manage the projects that should be realized in the short term while respecting the future of the enterprise.

ACKNOWLEDGEMENTS

This paper has been written as part of the 'SAP-leerstoel'-project on 'Extended Enterprise Infrastructures' sponsored by SAP Belgium.

REFERENCES

Ambler S. (2002). Architecture and Architecture Modeling Techniques. Retrieved from http://www.agiledata.org/essays/enterpriseArchitectureTechniques.html (on 29/1/2003).

Berners-Lee Tim (2001). The Semantic Web. Retrieved from http://www.scientificamerican.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

Bernus, P., Nemes, L., Schmidt, G. (2003) Handbook on Enterprise Architecture, Springer-Verlag, pp 778.

Boehm B., *Software Engineering Economics*, Prentice-Hall Englewood Cliffs, 1981, pp 767.

Borgatti S., Organizational Theory: Determinants of structure, 2001, Retrieved from http://www.analytictech.com/mb021/orgtheory.htm (on 4/3/2003).

BPMI.org, http://www.bpmi.org/bpml.esp.

Business Process with BPEL4WS: Understanding BPEL4WS, Part 1. Retrieved from http://www-106.ibm.com.

CIO Council, 1999. Federal Enterprise Architecture Framework version 1.1, pp 80. Retrieved from www.cio.gov/documents/fedarch1.pdf (visited on 29/1/2003).

CIO Council, 2001. A Practical Guide to Federal Enterprise Architecture. Retrieved from www.cio.gov/documents/bpeaguide.pdf (visited on 29/1/2003).

Clements P., Kazman R., Klein M., 2002. *Evaluating software architectures*, Addison-Wesley, pp302.

Cook M., 1996. Building Enterprise Information Architectures, Prentice-Hall, pp 179.

DAML, http://www.daml.org/.

Department of the Treasury, Chief Information Officer Council, July 2000. Treasury Enterprise Architecture Framework, Version 1, pp. 164. Retrieved from http://ustreasury.mondosearch.com/cgi-bin/MsmGo.exe?grab_id=49270800&EXTRA_ARG=IMAGE

Frankel D., Parodi J., 2002. Using Model-Driven Architecture to Develop Web Services. Retrieved from http://www.omg.org/attachments/pdf/WSMDA.pdf (visited on 29/1/2003).

Farooqui K., Logrippo L., de Meer J. (February 1996). The ISO Reference Model for Open Distributed Processing – An Introduction. Retrieved from http://lotos.site.uottawa.ca/ftp/pub/Lotos/TechRep/CNIS-94.pdf.

Frankel, D., Parodi, J. (2002). Using Model-Driven Architecture to Develop Web Services, IONA Technologies white paper.

Goethals F., Lemahieu W., Vandenbulcke J., 2003. Identifying Web Service Integration Challenges, *IRMA conference proceedings*.

Goethals F., Vandenbulcke J., Lemahieu W., 2004a. Developing the Extended Enterprise with the FADEE, *ACM SAC 2004 conference proceedings* (to appear).

Goethals F., Vandenbulcke J., Lemahieu W., Snoeck, M., De Backer, M., and Haesen, R., 2004b. Communication and Enterprise Architecture in Extended Enterprise Integration, ICEIS 2004 conference proceedings (to appear).

Hilliard R., Impact assessment of IEEE 1471 on The Open Group Architecture Framework. Retrieved from www.opengroup.org/architecture/togaf7/ procs/p1471-togaf-impact.pdf (visited on 29/1/2003).

IFIP-IFAC Task Force, 1999. GERAM: Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3. Retrieved from http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/GERAMv1.6.3.pdf (visited on 29/1/2004).

Koch C., November 15, 1998. Put IT in writing, *CIO Magazine*. Retrieved from http://www.cio.com/archive/111598/sla_content.html (visited on 29/1/2003).

Kruchten P. (November 1995), The 4+1 View Model of Architecture, *IEEE Software*, pp. 42-50.

Lassing N., Rijsenbrij D., van Vliet H., September 2001. Zicht op aanpasbaarheid, *Informatie*, pp 30-36.

Linthicum D.,2000. *B2B Application Integration: e-Business-Enable Your Enterprise*, pp 464, Addison Wesley.

Maier M., The IEEE 1471-2000 Standard - Architecture Views and Viewpoints. Retrieved from www.opengroup.org/architecture/togaf/agenda/ 0107aust/presents/maier_1471.pdf.

Parker K., (7/2001) Collaborative Manufacturing in the e-Business era. Retrieved from http://www.manufacturingsystems.com/custompub/0701qad.pdf (on 29/1/2003).
PIP 2A4, Query Sales Promotion and Rebate Information, RosettaNet. Retrieved from http://www.rosettanet.org.
Podolny J., Page, K., 1998. "Network forms of organization." ARS 24 (1998):57-76.
Pollock J., 2002. Dirty Little Secret: It's a Matter of Semantics. Retrieved from http://eai.ebizq.net/str/pollock_2a.html (visited on 29/1/2003).
Soni, D., R.L. Nord & C. Hofmeister (1995). Software architecture in industrial applications, in: R. Jeffrey, D. Notkin (eds.), *Proceedings of the 17th International Conference on Software Engineering*, ACM Press, pp. 196-207.
Sowa J., Zachman J. (1992). Extending and formalizing the framework for information systems architecture, *IBM Systems Journal*, Vol. 31, No. 3, pp. 590-616.
Tapscott D., Caston A. (1993). The New Promise of Information Technology, McGraw-Hill, pp. 313.
The Chief Information Officers Council (September 1999). Federal Enterprise Architecture Framework Version 1.1, pp. 41.
The DAML Services Coalition (2001). DAML-S: Semantic Markup For Web Services, http://www.daml.org/services/daml-s/2001/10/daml-s.html.
The Open Group, Architecture Description Markup Language (ADML). Retrieved from http://www.opengroup.org/architecture/adml/adml_home.htm (on 29/1/2003).
TOGAF (December 2002), The Open Group Architecture Framework (TOGAF), Version 8, Enterprise Edition, pp. 303.
UDDI.org, www.uddi.org.
van de Heuvel W., Proper E (November, 2002). De pragmatiek van architectuur, *Informatie*, pp. 12-16.
van der Lans R., 2002. Web services voor EAI en B2B. Workshop SAI, 14/10/2002.
Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl.
Zachman J.,1987. A framework for information systems architecture, *IBM Systems Journal*, Vol. 26, No.3, pp 276-292.
Zachman J. Enterprise Architecture and Legacy Systems, Getting Beyond the "Legacy", http://members.ozemail.com.au/~visible/papers/zachman1.htm.

---

[i] Parts of this paper are extracts from the conference proceedings of the ACM SAC 2004 Conference (Goethals et al., 2004a) and of the ICEIS 2004 conference (Goethals et al., 2004b).

[ii] Consequently there are two communication gaps. This may seem evident, but neglecting these communication gaps lies at the basis of a substantial number of project failures.

[iii] The problem we have tackled is fundamental and is all too often neglected (especially by IT people)! The fact is that companies do not like other companies to make reservations of which the confirmation only depends on the intentions of the company making the reservation. The commitment that should be part of the reservation is actually no commitment at all as the confirmation only depends on the wishes of the other party!

[iv] Of course, composites may be made from the Zachman primitives.

[v] UML is often used in design and development tools. XMI (XML Metadata Interchange), which can be seen as a way to save UML models in XML, could play a role in the lifecycle interoperability matter. Note that the adequacy of UML for supporting architecture semantics itself has been widely questioned (The DAML Services Coalition, 2001).